A web interface for a web query synthesiser

By

Sunil Sunny Jacob, B.E.

A dissertation submitted to the
School of Computing
in partial fulfilment of the requirements for the degree of

Master of Computing

University of Tasmania

December 2006

Declaration

I, Sunil Sunny Jacob, hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma in any tertiary institution, and that, to the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Sunil Sunny Jacob

Abstract

Synthesised queries have been found to perform better than user formulated queries in retrieving relevant documents from the web. This thesis aims to prove that it is possible to perform real time synthesis of queries based on user feedback for interactive web searches and integrate the system with a web browser for seamless user experience. A system is developed that implements the algorithms outlined in previous research and the results obtained by experiments conducted using the system is compared with the original results. The system is able to show similar search performance, measured by precision and recall, but now performs this in real time.

Acknowledgements

I would like to thank my supervisor, Dr. Vishv Malhotra, for providing timely guidance and support towards completion of this work,

Prof. Bala Srinivasan for passing on his enthusiasm for life and research, Dr. Byeong Kang for inspiration to do research and timely advice,

Fellow postgraduate students, especially David Johnson, for providing access to his research, useful feedback and technical expertise, Mathew Osbourne, for providing support network and a friendly ear over the course of the year,

Colleagues at Dytech who were always interested to know about the state of my research, especially my manager, Dave Furlani, who has allowed me flexibility in work hours and motivated me to do my best,

My friends in India and Hobart, who have been supportive and understanding, though I was virtually "invisible" for a year,

My family, who though far away, have constantly stood by and encouraged me,

And finally to God almighty, who makes everything possible.

Table of contents

List of tables

List of figures

1. Introduction

The World Wide Web (WWW) provides access to a wide range of information resources in the form of text documents and a wide variety of visual and audio formats. The freedom from traditional technological constraints in publishing content has resulted in a variety of information sources for topics that may vary from story books to scholarly databases. With the ever increasing number of information items being added, it has become increasingly important to correctly search for the most relevant information.

Many web technologies have been developed to assist in finding information on the Internet. Web search engines, for example, assist users to find information on the Internet by providing matching results to a search query. A search typically involves a short query from a searcher consisting of a few keywords that expresses their information need. The search engine locates documents carrying these keywords. An unorganized list of these documents would be hardly useful to the searcher. Search engines match the located documents against the query words to rate the relevance of the document. Searchers perceived information need is expressed through the keywords in the query. The ranking of search results before presenting them to users makes it easier to find the topic of interest.

Search engines require the users to type in queries that best represent their current information need. This involves selecting words and combining them in a form acceptable to the search engine. They may make use of web query operators like AND, OR, NOT to better define relationships between words. Users, in most cases, are able to select appropriate words to form a query and retrieve reasonably good results from the search engine. Good performance in most cases, however, does not guarantee that every information need can be satisfied easily. Searchers occasionally do encounter difficulties in completing successful searches for their

information needs (Leroy, Lally & Chen 2003). Some of the challenges faced are as follows:

- Limited knowledge of the search domain or area of search makes it difficult to choose words for a query that best expresses the users information need.
- Inability to find highly specific information regarding the topic of interest, though resources on the general topic are easily available.
- Looking for multiple high-quality sources of information regarding the same topic is often difficult.

Since the search engines can only offer search results based on the query presented, the search results returned will have few relevant results if the query does not cover the information need correctly. In such situations, users often resort to modifying the query, in response to the presented list of search results, to better align the query to their needs. This approach takes time and users are often frustrated by the multiple modified queries required to obtain relevant results (Brin & Page 1998). Human efforts to improve queries are limited to trial and error as it is very difficult to fully predict the effect of a change in a query on search performance.

Query modification techniques can be used to suggest better queries for use by search engines. In particular, relevance feedback on search results was found to be effective in improving the search results in small static collections (Harman 1992). Extending this feedback approach to web search results could help in improving their quality.

When using relevance feedback on web search results, considering all documents returned by the search engine for query expansion is not feasible. Instead by providing relevance feedback for the top N (where N is typically 10, 20) documents, there would be sufficient data available to produce a synthesised query. Patro (2006) investigated whether performing query synthesis using user feedback on an initial search result could produce better queries. Her user survey has shown that synthesised queries performed better than user modified queries.

The functional ability to construct queries is but one aspect of a tool that can help human searchers to search for their information needs on the web. There are a number of non-functional specifications that need to be met if we are to provide a tool that searchers could use. Included among these specifications are:

- Real time performance of query synthesis process.
- User friendly interface for obtaining feedback from the searcher to understand their information needs.

The goal of the current work is to design a real-time query synthesiser that is well integrated with popular web browsers and continues to provide performance improvements to the reported query synthesis algorithm.

In addition to reporting the design and implementation of the Real-Time query synthesiser, evidence of the performance of this tool is provided. To this goal, the Real-Time query synthesiser is used to search for information needs defined in Patro's user survey and show that matching performances are obtained in timeframes acceptable to interactive searching.

1.1. Thesis structure

The thesis is structured as follows:

Chapter 2 presents a literature review discussing relevant and related research in information retrieval: how web information retrieval is different from the traditional information retrieval; user characteristics when interacting with the search engines; suitability of information retrieval models for web retrieval; types of web search tools and forms of query modification.

Chapter 3 details the system architecture of the Real-Time query synthesiser and describes the implementation of the system. A description of existing software components, available on the web that was used in the system, is also included.

Chapter 4 presents the hypothesis, and the intermediate steps identified to validate the hypothesis. Results recorded using the software is presented followed by a

discussion of the results, and comparison with previously reported performance indicators.

Chapter 5 concludes the thesis by defining scope for future work and providing a summary of work done.

2. Literature review and motivation

Retrieving information from computer systems has been extensively studied by computer scientists over the past several decades (Broder 2002). During that time, the advances in computer hardware have increased the potential for information retrieval applications and allowed for much wider variety of information content to be indexed and searched. Today this field of research has been extended to include the Internet, where documents reside on computer systems around the world, and this presents a new set of challenges and possibilities.

To better understand the issues and their relationships to each other, the discussions in this survey are organized on a theme basis rather than on the individual research group or paper.

This chapter is organized as follows: Section 2.1 discusses the basic concepts of information retrieval. Section 2.2 discusses the newly emerging field of web information retrieval and what properties make it different to the traditional information retrieval. Section 2.3 discusses the suitability of information retrieval models for use in web information retrieval. Section 2.4 presents two common web search technologies: directory services and web search engines. Section 2.5 presents query modification and its different forms. Section 2.6 presents previous research done by Sunanda Patro to produce synthesised web queries. Section 2.7 concludes the chapter.

2.1. Information retrieval

Information retrieval, as defined by (Baeza-Yates & Ribeiro 1999, p. 1), refers to the

"representation, storage, organization of and access to information items".

Information retrieval research was initially concerned with indexing, storage and retrieval of information from small bibliographic databases in libraries (Van Rijsbergen 1986). The data in these systems later expanded to include keywords and abstracts. The success of these initial systems led to development of similar systems for specialized databases - medical systems like MEDLINE (Chung et al. 1999).

From the field of information retrieval, few main concepts are explained next. Terms refer to the words that function as keywords when looked up in a retrieval system. These terms may be combined together to form queries by simply placing them together or by connecting them using query operators. A query thus formed, may be used to represent the users information need (Broder 2002).

The information that the user looks for when interacting with a retrieval/search system is called the user's information need (Brin & Page 1998; Broder 2002). Often users are unable to accurately specify their information need. This could be due to a variety of reasons like their limited domain knowledge to form queries, inability to structure the terms in the query to get relevant results etc. This gap between the users information need and the keywords employed by the user to represent that need is referred to as semantic gap (Zhao & Grosky 2002).

2.2. Web information retrieval

Searching for information on the web is significantly different from traditional information retrieval (Lawrence & Giles 1998). The dynamic nature of the content on the internet and high update frequency presents a whole new set of challenges not faced by traditional information retrieval systems. The characteristics of web based retrieval that differentiate it from the traditional information retrieval are (Chowdhury 2004, p. 331):

- Content on the Internet is distributed and generally does not conform to set standards. Often duplicate information may be stored at multiple locations, making the task of retrieving and displaying information more complicated.

- Format of content may vary widely - text, video, audio, databases, images, graphs etc.
- Deep web refers to the set of documents that require authentication to be viewed and may or may not be available for indexing. On the other hand, surface web is the publicly available information that can be indexed and searched on. Access permissions and copyright issues have to be considered when parsing and extracting information from such data sets.
- Difficult to assess the quality of information published on the internet, due to lack of quality control mechanisms.
- Frequency of changes are higher – web searches have to deal with changes over millions of pages, also track movement of information between servers
- Language support is also necessary – different alphabets and often of very large sizes like Chinese and Japanese Kanji.
- Volume of information on the Internet that can be searched is much higher

The issues encountered in web information retrieval discussed above are heightened with the varied profile of users that use web search systems. Several surveys have been conducted to study the characteristics of web search users (Jansen et al. 1998 ; Kobayashi & Takeda 2000; Muramatsu & Pratt 2001; Ruthven & Lalmas 2003). The search user behaviour was studied through the use of transaction logs provided by the AltaVista search engine by Spinck and Jansen (2004). Their findings were as detailed below:

- The mean search query length employed by users is about 3 terms and slowly increasing. During the earlier years, single query terms were more widely used. Later on, as the amount of information returned by the search increased, users had to resort to more query terms to better represent their information need.
- About 15% of all queries used query operators, of which more than half consisted of the PHRASE operator and most of the remaining using AND operator. This shows the general reluctance of search users to use advanced features of the search engine to formulate their queries. Basic training in

Boolean logic has shown to produce an increase in users performance in obtaining relevant results (Lucas W. & Topi H. 2002)

- Web search users view on average two to three documents from the result of their query, before either giving up or reformulating the query. Also more than half of the users glance through only the first page of results, and rarely move to the second or third page.
- Interactions between searcher and search engine were relatively simple compared to traditional IR systems. The majority of web searchers were not aware of and/or did not use the sophisticated features provided by the search engine, whereas users of traditional IR systems made efforts to learn and use advanced features.

With such a varied user base, it is difficult to tune the system in response to user characteristics. The representation of the information need can be done using information retrieval models as described in the next section.

2.3. Information retrieval models used in web search retrieval

Traditional information retrieval research has led to the development of various models that have had different characteristics and degrees of success in retrieving information. Often these models are used in conjunction with each other to perform better in certain information retrieval systems. For example adding terms weights to the Boolean model, substantially improves the search results (Van Rijsbergen 1983). The suitability of these models for web search retrieval is investigated here.

2.3.1. Boolean model

This model uses logical operators like AND, OR and NOT in query terms to represent the users information need. Its simple operation has made it a highly successful model for many commercial implementations (Lucas W. & Topi H. 2002). In spite of its widespread use in search engine technology, the Boolean model has its limitations (Salton 1984). The clear demarcation of search results into relevant and non-relevant categories based only on the presence/absence of query terms has meant that the users have had to restructure the query to manipulate

whether a desired document appears or not. Term weighing has proven to be an effective mechanism to extend the Boolean model (Salton, Fox & Wu, 1983).

2.3.2. Vector space model

The vector model was first proposed by Salton (1971). It tries to overcome the limitation of the Boolean model by making partial match of terms possible and uses two vectors – the document vector and the query vector. The model aims to bring a convergence between these vectors in the form of a similarity function. Term-weighting has been used to indicate the frequency of appearance of a term, or its location in a document. These weights can then be used to compute the degree of similarity between documents and the query posted. Though term-reweighing improves the relevancy of results, significant increases can be established only with the use of query expansion or relevance feedback, as argued later on.

2.3.3. Probabilistic model

The probabilistic model was first introduced by Robertson et al (1981) and estimates the probability that a given document will be relevant to the user given a particular query. This model differs from the vector space model with the introduction of probability matching function instead of the similarity measure.

2.3.4. Extended Boolean model

Salton (1984) proposed the Extended Boolean model in an attempt to combine the features of the Boolean and the vector space models. It allowed the use of structured queries and provided a mechanism to reduce the impact of Boolean operators. Both documents and queries have weights calculated based on normalization techniques, which can then be used to rank documents. Due to the increased computational cost involved, it works well with small document collections.

2.3.5. Cluster model

This model group documents, that satisfy a set of common properties, into clusters (Liu & Croft 2004). It is based on the assumption that similar documents will

contain the same information. This method involves training queries that retrieve documents from a particular cluster, with each cluster weighed to indicate the number of relevant documents returned from it. The construction and maintenance of cluster information introduces more computational cost and are employed by search engines to group results from the same website.

Of the models discussed, the Boolean model is most suited for web information retrieval systems. Search engines accept Boolean queries and feature web query operators like AND, OR in their advanced search options. By using query expansion along with the computed use of Boolean web query operators, the search results returned by the queries can be improved.

2.4. Types of web search

Searching for information on the web can be done in several ways. The increased need to search for information has resulted in the development of two distinct service types: Directory service and Search engines.

2.4.1. Directory service

Directory services like those provided by Yahoo! structure information into categories that are decided by experts in those fields. The level of granularity of topics is important, and the users often have to browse through multiple headings before locating the topic of actual interest. This can be an issue when the user is not aware of the top level topic name and unaware of how to eventually reach the topic that they are looking for. Listed under these topic headings are the actual resources that would satisfy the users' information need. Also the users can search through the topic headings if they wish to locate a topic of interest.

Directory based search has not been very successful in locating information items of interest to the user (Bruza, McArthur & Dennis 2000). The requirement placed on the user to have sufficient knowledge about the topic hierarchy works well for certain advanced search users, like scientists looking for information in specialized domains. The web search user, on the other hand, is often unsure of his information need and thus may not be able to locate the topic from the given directory.

2.4.2. Search engine

Search engines employ a different approach to the directory service to meet the users need. Instead of searching through the whole text of the information resource, the words are indexed and searching usually involves traversing this index to find the terms of interest. Commercial search engines typically do not reveal their architectural details. Arasu et al (2001) discusses web search engine design and introduces a generic search engine architecture.

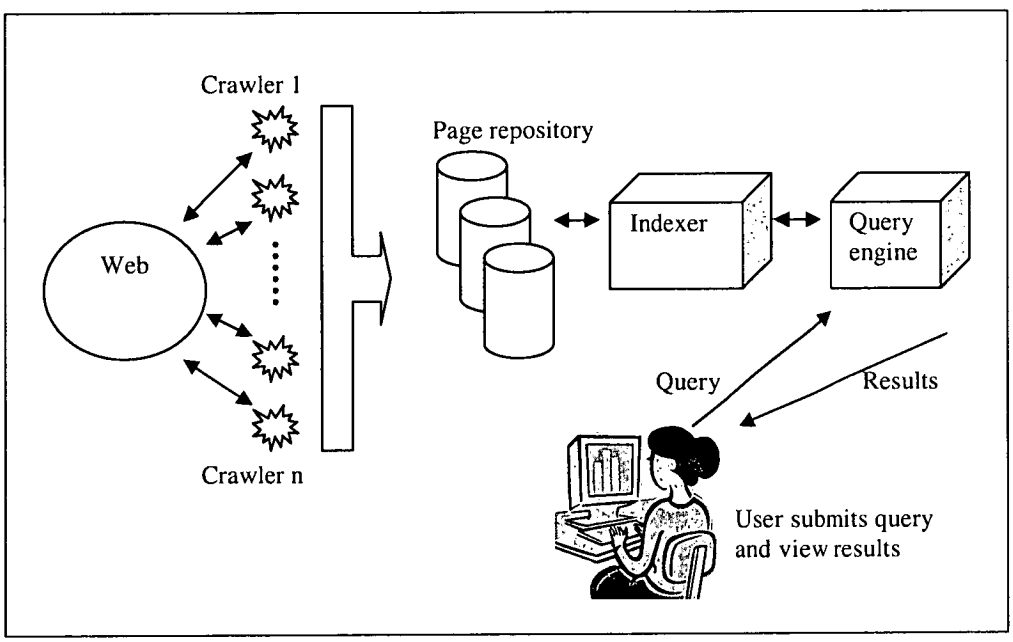

Figure 2-1. Search engine architecture (Spink & Jansen 2004)

The crawler travels the Internet, capturing web page data, and sending the information back to the page indexer. It is provided initially with a set of URLs whose pages it first extracts, and later on, depending on the usefulness of the links on those pages, investigates them further. The pages extracted by the crawlers are stored in the page repository. The indexer extracts words from these pages that are then used to form a searchable index. The requests from the users are handled by the query engine, which will need to use the indexer and page repository to adequately return resources that match the user query. Due to the size of the Internet and the varied content it hosts, the search engines have to be designed to

handle huge amounts of data and simultaneously respond to large number of requests

2.5. Query modification

The query specified by the user will return the result set, using terms in the query as keywords to look up in the document collection. The ranking procedure conducted on these results will attempt to give a higher ranking for the documents that match the query specified. Van Rijsbergen (1986) spoke of the limits of providing increasingly better ranked results based only on the initial query, and indicated the need to modify that query to match more relevant documents. This is particularly important when incorporating relevance feedback information from the user to better represent the information need. Feedback information can be incorporated into the query with help of query expansion and term reweighing of the initial query.

2.5.1. Query expansion

When the feedback iterations are completed, the system may be able to add some additional words to the initial query that would indicate the searchers information need. This "query expansion" would be possible through the use of query operators or by simply appending additional query terms to the existing query. The idea here is that using additional relevant terms from the search domain will increase the relevancy of results returned. The usage of query expansion has been shown to improve document ranking (Alemayehu 2003). Query expansion has also been suggested to improve precision in search results (Bharat & Henzinger 1998).

2.5.2. Term reweighing

This technique assigns weights to the terms in the query to better indicate their relative importance. In the Boolean information retrieval model, the weights are either true or false indicating the presence or absence of the terms in the document. Later the vector space model proposed a finer grained weighing to indicate the importance of terms when representing the need (Salton 1984). As specified earlier, term-reweighing in conjunction with query expansion yields greater improvement in ranking than either one can provide individually (Harman 1992).

2.5.3. Relevance Feedback

The use of relevance feedback as a query reformulation strategy has been found to be effective in a number of cases. Salton (1971) used this technique in the SMART system and found that it provided good improvements in precision on small test collections.

In a relevance feedback cycle, the user is presented with a list of documents based on an initial query and is then prompted to mark the relevant documents from this list. In doing so, the first ten (or slightly higher number) ranked documents may be marked as relevant or irrelevant. The system then resolves which terms or characteristics to select from the marked relevant documents to better represent the users information need. Van Rijsbergen (1983) outlined the desirable characteristics of relevance feedback mechanism when used in information retrieval systems:

- The system may not be able to completely judge if the retrieved document is relevant or not as the same query run by different users may select different set of documents as relevant according to their particular information need. Hence there is a need for feedback to confirm the information need.
- The system should not use the knowledge accumulated by one user to understand the information needs of other users. A user centric understanding may be developed during the iterative feedback process.
- Systems must reflect the properties of the documents or objects in the collection and leave the determination of usefulness of its ranking to the users.

The main advantages of relevance feedback over other query reformulation mechanisms (Baeza-Yates & Ribeiro 1999, p. 118) are as follows.

- Relevance feedback shields the user from the details of the query reformulation process, by enabling the user to provide relevancy judgment on documents and leaving the finer details to be worked out by the system.

- It breaks down the search process into more manageable relevance feedback iterations, that enables users to refine their information need as they interact with the system
- It offers a controlled procedure to emphasize terms that are relevant and de-emphasize them, if irrelevant.

Having multiple interactions with the system during a relevance feedback cycle helps the system to converge on the exact information need of the user (Van Rijsbergen 1983).

2.6. Sunanda Patro's research

Sunanda Patro (2006) investigated the search performance when using query synthesis on web search results. She noted in her research that though users had difficulty in expressing their information need, they were better equipped to assess the relevance of presented search results. The algorithm she developed was able to produce synthesised queries using relevance feedback information from users. The algorithm is explained in more detail in Section 3.2.5.

2.6.1. Survey

The survey conducted by Patro involved forming best-effort queries for a list of 25 topics, with each topic having a detailed description to maintain a consistent information need. The time taken by experienced web search users to form queries, which returned good results for the information need, in the survey was found to be 24.6 minutes on average (see Table 4-4). The search performance of the synthesised queries, as measured by precision and recall, were better than the user modified queries.

2.7. Summary

Web searches performed today are able to satisfy most users' information need. When this information need is not easily expressible, relevance feedback is proven to be an effective tool to assist the user to form queries. Performing relevance feedback on search results in an interactive manner can help users evaluate the

effects of their decisions and will be a useful component addition to existing search engines.

3. Methodology

In the previous chapter, the motivation and past research were described. The research by Sunanda Patro has proven that synthesised queries can perform better than user modified queries. However it has not been shown yet that the synthesis can be done through a user friendly interface with delays acceptable to web searches. Our goal is to achieve these aims. To compare with the search performance metrics reported in Patro's survey, a software system, Real–Time query synthesiser, is developed to integrate with the web browser.

The system would verify that real time performance can be obtained when performing query synthesis based on user feedback. In order to do this, the system must be capable of the following:

- Present an intuitive interface to receive feedback and perform certain pre-processing concurrently with feedback entry
- Store the relevance feedback information efficiently in memory and in a form easily accessible for query synthesis
- Perform query synthesis in an acceptable timeframe and present the results

It is useful to provide a sketch of the user session to understand the internal working of the implemented software system. The following steps describe a typical user session:

a. Given an information need, the searcher initiates a session by providing an initial search query to the Real-Time query synthesiser.

b. The Real-Time query synthesiser obtains a list of matching links from the Google search engine. This list of links is the same as the links received when the searcher gives the query to Google directly.

c. The synthesiser displays 10 links in a search results page to seek searcher feedback.

d. The searcher provides feedback related to each displayed link through the feedback radio buttons attached to the link snippet. The feedback can be provided by viewing the snippet or after viewing the related document in a browser. Though we do not intend to place any limit on the number of documents for which feedback is required, we found that feedback on about 20 documents is most suited.

e. User initiates a new search using synthesised query by pressing Recompute button. This causes the Real-Time query synthesiser to form query using the given feedback and sends this query to Google for search.

f. Results obtained are displayed to the user in a form similar to the initial search query results described in step b. The searcher can provide further feedback if desired.

This chapter is mainly concerned with the design and development of the system to achieve the goals outlined previously. Performance and other related issues will be discussed in Chapter 4. Section 3.1 describes the survey of existing tools suitable for integration with the system. Section 3.2 outlines the system architecture of the Real-Time query synthesiser and explains the components in detail. Section 3.3 provides a summary of the chapter.

3.1. Survey of external tools

In order to achieve the above goals set out for the software, a brief survey was conducted of the existing tools that could be used to perform specialized functions within the system. The observations from this survey are detailed below:

3.1.1. Google SOAP Search API

An API (Application Programming Interface) provided by Google was considered for systems development. The API called Google SOAP Search API (Google 2006) provides access to web search results through SOAP (Simple Object Access Protocol), which is a protocol for communicating XML (Extensible Markup

Language) messages over computer networks. The API provides detailed instructions for developers to incorporate search functionality into their existing application in either .Net or Java.

The advantages of using an API for communicating with the search engine are numerous. The information obtained from search engines follow a standard format and can be easily integrated into the existing system. Changes to the service would ensure backwards compatibility, in most cases, with adequate technical support available through online forums.

In case of Google SOAP search API, the search process was of a different nature to the standard Google web search, as accessed on a browser. The following observations are relevant:

- In current service standards, the API returns inferior results when compared to web search. The API has provisions for providing results for periodic automated queries and this could explain the reason for inferior results. The search results returned by the API for a particular query were often a small part of the total search results returned on the web version.
- Time taken for search results to be displayed is greater than those experienced with web search. This could be attributed to the larger message size (XML) compared to HTTP messages. Also, since the API is in beta stage, it is possible that it receives lower priority.

Due to these difficulties, the above API was not considered during systems development. The alternative approach decided on was of retrieving information as standard HTML web document from the webpage based on identifiers in HTML (see Appendix 4)

3.1.2. HTML Parser

When documents are marked as relevant or irrelevant through user feedback, the words present in these documents need to be extracted and stored within internal data structures. For this purpose, the parsing of HTML must occur in real-time.

Documents available on the internet follow a myriad of different formats for presenting information (Chakrabarti 2003). This meant that attempting to extract the words from these documents would require an understanding of the structure of web pages and the ways HTML is used by developers across the world.

The HTML Parser engine was considered for this task. The main reason for considering an existing engine was because the engine had been developed and extensively tested by a number of experienced web technologists. The HTML Parser v1.6 described is available under the GNU General Public Licence (*HTML Parser* 2006) and its salient features include extraction and transformation of HTML in real-time. The parsing tool could be easily integrated into the system as a JAR (Java Archive) file.

The extraction of words from web pages of varying qualities was tested. The textual contents of the pages were adequately covered and the words returned by the parsing engine were as expected. This coupled with the ease of integration, prompted the decision to include the HTML Parser as a module during systems development.

3.1.3. Stopword removal

Search engines ignore certain frequently occurring words in queries in order to provide good search performance. These words, called stopwords, do not improve the description of the information need and can be removed from list of words considered for query synthesis.

For stopword removal, the tool designed by Martin Hassel (2006) was used. The tool, part of a package for indexing and storing words for text summarization, was able to remove stopwords from a list of words in file (see Appendix 3) and provided the ability to perform this operation on a huge list of words in an efficient manner.

3.1.4. Removal of common words using stemming

From the search results returned by the initial query, it was observed that variations of certain words were among the words extracted. For example, the words

"algorithm" and "algorithmic" were extracted for a search on the query "software". This meant that more than one word that are almost similar, may be potentially considered for query synthesis.

Similar words often stem to the same word. For example, "algorithm" and "algorithmic" stemmed to the word "algorithm". Using this information, it is now possible to select the word that is most likely to perform well if added during query synthesis. This estimate is based on the frequency of the word within the document, with the most frequent variation of the stemmed word being selected.

Porter's algorithm (Porter, M. F. 1997) was used for stemming words. A Java implementation (Porter, M. 2005), written by the original author of the algorithm, was tested and selected for integration as a module within the system.

3.1.5. Browsing module

The web pages associated with the search result, can be viewed in a separate browser window. This may be needed when the search summary for a particular result does not provide sufficient information to decide its relevance for feedback. The browser module opens the URL retrieved from the search result in the default browser of the system. To allow the URL to be opened in an external application, system permissions need to be set during execution of the program. The module integrated into the system allows for the selection of the default browser in opening web pages in an application external to the core system.

3.1.6. Google search engine

In the previous survey reported by Patro, the search engine from which the results were gathered was Google. Since the search performance is to be compared with the previous reported values, the Google search engine was selected for the current system for performing all searches. This would ensure that the results generated using the initial queries match closely with those reported in the previous survey.

3.2. System Architecture

In the previous section, the modules that are to be integrated into the system were presented. The overall system architecture is described in this section followed by a description of the query synthesis process in the current system. The system has been implemented in Java mainly to achieve real-time performance using concurrency features of Java. The concurrency would allow many tasks to be accomplished while user is browsing the displayed information. Java is also platform independent and provides customizable user interfaces, adding more flexibility to the system.

The main components of the system are as follows (shown in Figure 3-1):

- Graphical user interface
- Browser module
- HTML Parser module
- Word pre-processing engine
- Internal data structure
- Query synthesiser

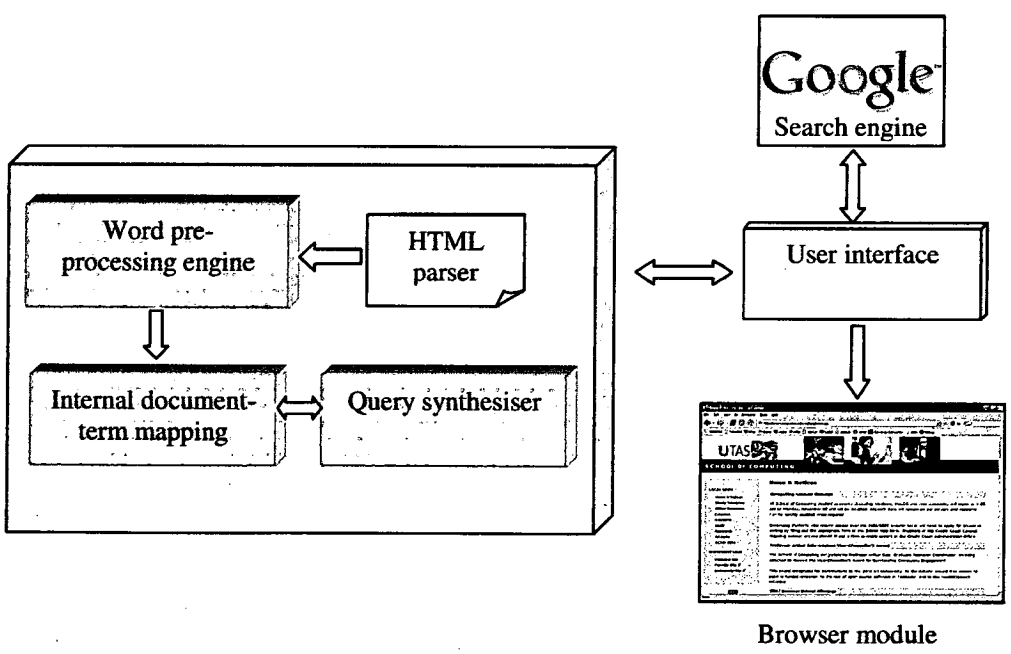

Figure 3-1. System Architecture of Real-Time query synthesiser

The main interactions of the components in the system are as follows (Figure 3-1):

- The initial query is entered on the user interface, and search results for the query displayed from the search engine.
- When feedback is indicated against the displayed search result, the associated web page is parsed and the words extracted are sent to the Word pre-processing engine. These are then stored in internal data structures (document-term maps) for use by the query synthesis algorithm.
- When the process for query synthesis is commenced by the searcher by clicking on the Recompute button (see Figure 3-3), the algorithm works on the words in the document-term maps and produces a synthesised query.
- The synthesised query is sent to the Google search engine to retrieve the new page of search results and displayed to the search user.

The following sections explain the main components and the processes that occur between modules.

3.2.1. User interface

The user interface has been designed based on the search interface design made famous by Google (Sinha 2005). The interface was selected because it is widely used and displays essential information of interest to the web searcher in an uncluttered format. This design was extended to include feedback radio buttons for recording user feedback. The decision to align the feedback buttons to the right of the search results was driven by need to clearly partition the search results and their feedback. The user is able to quickly review the feedback entered at the end of the search session with greater ease when positioned away from the search body.

Figure 3-2 shows the user interface for initial query entry.

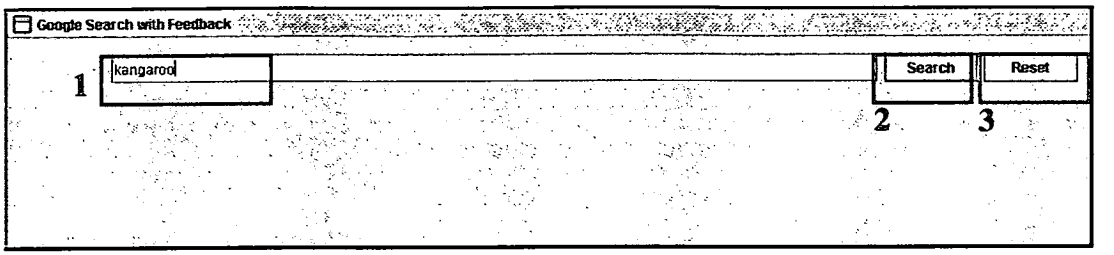

Figure 3-2. User interface for entering initial query

The user interface, as shown in Figure 3-2, contains a text box for entry of initial query [box 1 in Figure 3-2]. The Search button [2] is used for initiating communication with search engine and loading the first page of search results. The Reset button [3] can be used at any point during the search session to clear the screen of the initial query or of any search results loaded as a result of communication with the search engine.

Figure 3-3 shows that the user interface changes as the search result summaries are retrieved from the search engine, using the initial query.

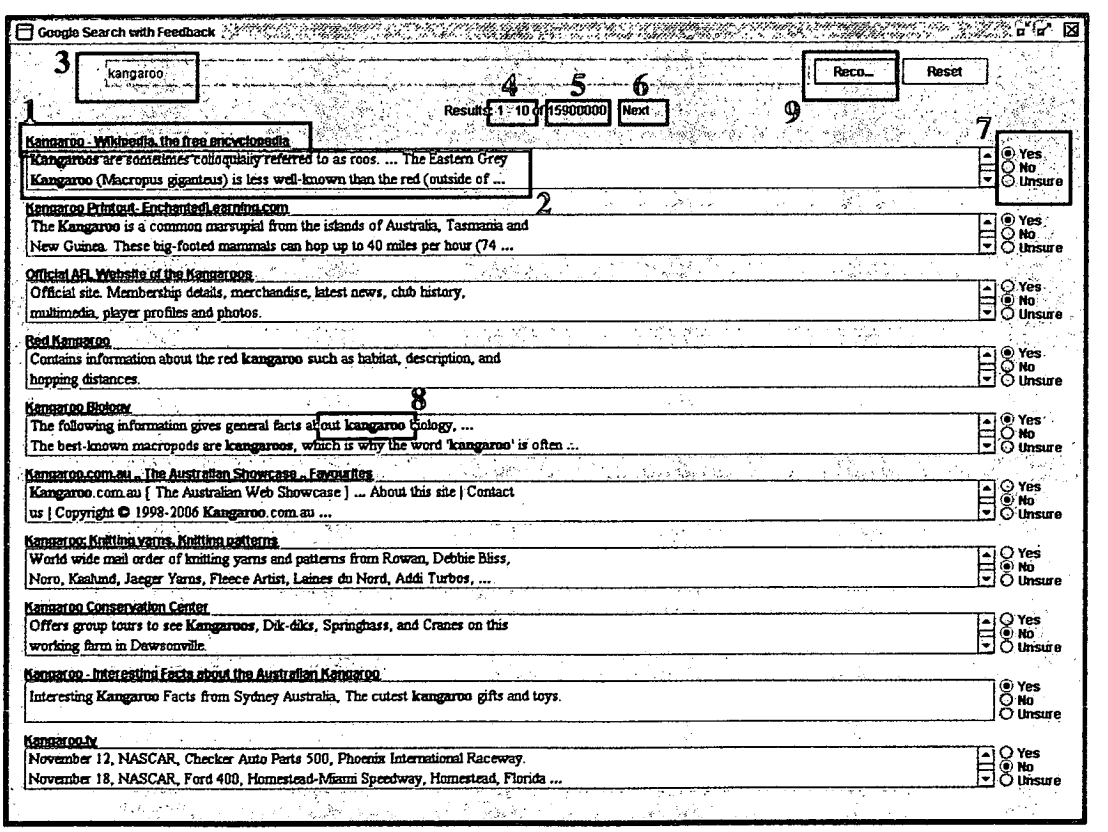

Figure 3-3. User interface after display of search results based on initial query

The display of search results, as shown in Figure 3-3, includes the title of the search [box 1 in Figure 3-3] and the search summary [2] for ten documents. These are retrieved based on the initial query entered in the previous screen (see Figure 3-2). The web page associated with the search result can be viewed in a browser by clicking on the search title [1]. The words of the search result highlighted by the search engine [8] are maintained in a similar format in this interface. The text box [3] for query entry is non-editable to show the query for the duration of the search session.

For each search result, the feedback options [7] available to the user are as follows:

- Yes - indicates that the search result displayed is, according to user evaluation, relevant to the information need
- No - indicates that search result displayed was not relevant or directly related to the information need.
- Unsure (default) - The user can indicate his uncertainty in search result evaluation by selecting this option. Selecting this option flags the query synthesiser to ignore this document entirely and is not included in the feedback process. For documents like PDFs that are not parsed by the system, this option is implicitly selected.

The number of search results matched is displayed as a range [4] along with the total number of documents matched by the query [5]. Links to the next or previous pages [6] are available as links based on the current page displayed. The Recompute button [9] is clicked to start the query synthesis.

Figure 3-4 shows the user interface after loading the search results based on the synthesised query.

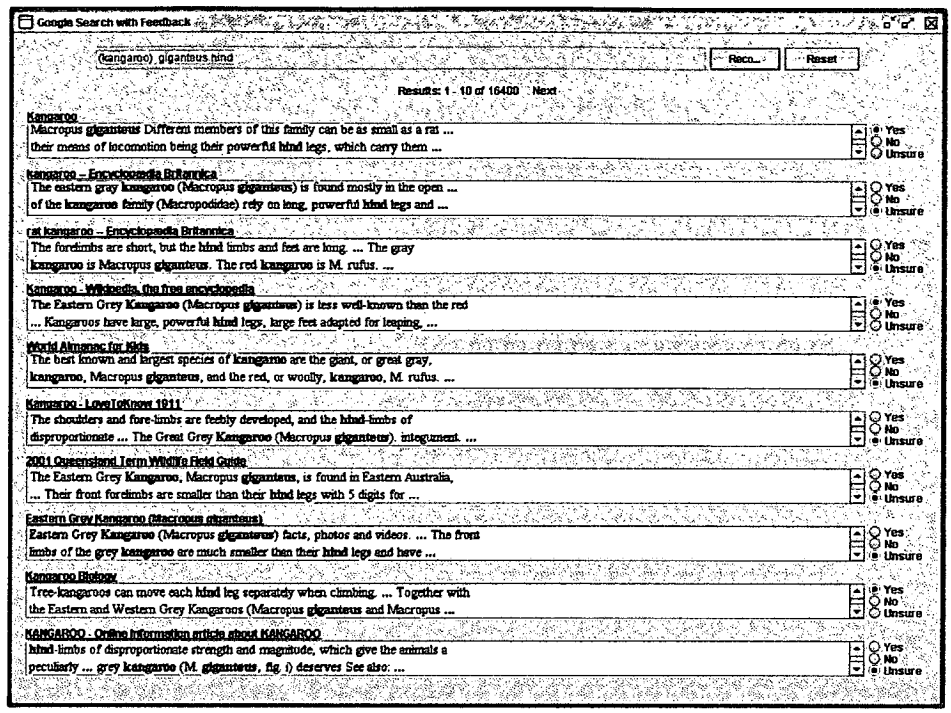

Figure 3-4. User interface after query synthesis

Figure 3-4 displays the new search results obtained from the synthesised query. The feedback radio buttons are placed alongside the search results in case additional feedback iterations are attempted. It also provides an indication of the feedback entered in the previous cycle, if the same document is displayed again. For example, results 1 and 4 were marked as relevant in the previous feedback iteration, and the synthesised query has retained them again in this round.

3.2.2. HTML Parser module

As relevance feedback is marked for a search result, the associated web page is retrieved and the words extracted. The HTML Parser performs parsing of HTML as a separate Java thread while the relevance of other search results is being decided. The parsing thread is started as soon as the user selects one of the feedback buttons (Yes or No) on a document. This is done to improve system responsiveness.

A custom HTML parsing module was added to extract the search results from the Google search page. This relied on identifiers that uniquely indicated the start and end of a search snippet, or search result. The ability to retrieve individual components of the search result like the title of the search, the URL of the

document and document summary was included. This is useful to correctly display as output to the user. At the search page level, the range of search results and the total number of search results matched was extracted. The complete list of HTML identifiers used to parse the search results are presented in Appendix 4.

3.2.3. Word pre-processing module

The words extracted from the documents, in their original form, cannot be used directly for query synthesis. This is because the words extracted may be too common (see section 3.1.3) or they may not be very effective in expressing the information needs (see section 3.1.4). The thread that performs the HTML parsing continues on to perform operations in the Word pre-processing module. To get the right set of words for processing by the query synthesis algorithm, the following operations are performed: basic processing, stopword removal and similar word removal.

3.2.3.1. Basic processing

The words extracted from the documents consist of letters and numbers. Since numbers are not required to describe most topics, they are removed at this stage. All the remaining words extracted from the document are converted to lowercase. This is done to maintain consistency within the system and to comply with the search engine restriction of using only lowercase words in the query (Google 2005).

3.2.3.2. Stopword removal

As described in 3.1.3, stopwords are ignored from the query by search engines when presenting search results. In the current system, the stopword removal is done soon after the extraction of words from documents using the same thread. The stopwords used in this module are listed in Appendix 3.

The stopwords are extracted from the file and stored in memory as a Java HashSet object (Sun Microsystems 2004c). The advantage of this approach is that the words are now represented by hash codes (primitive int wrapped in an Integer object) whose value is obtained using a hashing algorithm. Using Integer objects for

comparision (16 bytes) is much better than using Strings (40 bytes) (Roubtsov 2002).

3.2.3.3. Similar word removal

After the removal of stopwords from the words in a document, the thread now checks for the presence of similar words in the document. As discussed in section 3.1.4, similar words do not aid in improving the search performance. These words are identified among the words in the document based on their stemmed versions.

The module converts all words in the document into their stemmed forms and group words that stem to the same word version together. The frequency of unstemmed words in each group is then looked up and the word with the highest frequency (within the document) is selected as the representative from the group. The resulting list of words does not contain similar words and each word contributes to the improved performance of the query synthesis algorithm.

The information processed by this module is now ready for internal data representation as described in the next section.

3.2.4. Internal data representation

The pre-processing done on the words were discussed in the previous section. The efficient storage and retrieval of these words are now important for the query synthesis module. This is because the relationships between the documents and words need to be maintained completely in memory data structures and will be constantly referenced by query synthesis module. The main data structures used in Real-time query synthesiser are

- relevantHash and irrelevantHash
- docMap
- termMap

Next the above data structures are explained in more detail.

3.2.4.1. relevantHash and irrelevantHash

When the documents are marked as relevant or irrelevant, it is necessary to store the current feedback choice in a separate list. This would allow for frequent changes to feedback selection to be conducted without affecting the processed contents of these documents (as described in section 3.2.3).

Java Vector objects (Sun Microsystems 2004b) are used to store the expanding list of relevant and irrelevant documents. As discussed in section 3.2.3.2, hash codes (primitive int, stored internally as Java Integer objects) provide better search comparison efficiency than String objects. The documents are converted into their hash codes and stored in the relevantHash or irrelevantHash list based on the feedback indicated. If a change in feedback is required, the document hash code can be removed from one list and added to the other. This avoids the re-processing of the document for every change in feedback, as the processed document information is not lost, but only needs remapping. Next, the storage of feedback information at the document level is discussed.

3.2.4.2. docMap

docMap is the data structure used to hold the document-to-term relationship in the current system during the feedback entry process. Java Hashtables provide functionality to map objects as key-value pairs (Sun Microsystems 2004a). To record the relationship between documents and words (also called terms from IR research), the following actions are done for each document:

- The document's URL is converted into hash code and stored as a key in the Hashtable object.
- The words in the document along with their frequency of occurrence within the document are stored as values.

Table 3-1 shows the structure of the docMap data structure.

Key (document hash code)	Value (term, frequency)
1483455435	(kangaroo, 2)
	(animal, 1)
	(pouch, 1)
-416373960	(pouch, 2)
	(leg, 1)
323699380	....
	....

Table 3-1. docMap using Java Hashtable

In the above data structure, information regarding the document-term relationship is recorded as the feedback is marked. For example, the document marked by hash code 1483455435, has three words: kangaroo, animal and pouch with word frequencies of 2, 1 and 1 respectively.

The reason for storing the document level information is because the docMap structure is built dynamically during the feedback entry process. This means that multiple documents may be marked as relevant/irrelevant during the formation of the docMap data structure. During the feedback process, the user may decide to remove a document from feedback by marking the document using the "Unsure" radio button (see section 3.2.1). This would result in the whole document and the words in it to be removed from the data structure, which is more easily accessible when in document based format. For changes from relevant to irrelevant, or vice versa, this is not applicable as the document needs to be considered for query synthesis in both cases. In this case, the only change would be in the entries made in the relevantHash and irrelevantHash Vectors (see section 3.2.4.1).

Once the feedback entry process is completed, the docMap formed can be converted into termMap as detailed in the following section.

3.2.4.3. termMap

termMap uses Java Hashtables to document the relationship between the words (terms) and the documents, in a similar manner to docMap. This can be easily done because the document level information has been formed during the feedback entry process. Since the word "term" is more prevalent in information retrieval research, "word" and "term" will be used interchangeably from this point on

The formation of termMap is performed after the feedback entry is completed and before the query synthesis can begin. This change in state occurs when the "Recompute" button is pressed (see box 9 in Figure 3-3). To record term level information, the following actions need to be completed for each document in the docMap data structure:

- The words associated with the document are added onto the termMap as the keys
- For the value part of the key value pairs, an entry for the document is recorded against every word newly created in the termMap. The frequencies previously associated with the words in the docMap become associated with the documents in the termMap

The following Table 3-2 shows the structure of the termMap data structure.

Key (term)	Value (document, frequency)
kangaroo	(1483455435, 2)
	.....
	.....
pouch	(1483455435, 1)
	(-416373960, 1)
animal	....
.....	....

Table 3-2. termMap using Java Hashtable

When mapping from words to documents, multiple documents may have the same word. From the example shown in Table 3-1 and Table 3-2, the word "pouch" is present in two documents represented by hash codes 1483455435 and -416373960. In the docMap, these were the keys representing the documents. But in the termMap, the term pouch is the key and the documents, in which the term "pouch" was present, become the new values for this key-value pair. The frequency of the document 1483455435 for the term kangaroo is taken from the previous frequency 2 that was stored as value with the key kangaroo in docMap.

The information in the termMap is of more use to the query synthesis algorithm than of that in the docMap data structure. The repeated information (documents) in the termMap data structure has been compressed by the conversion to hash codes. The term level information in now ready for use by the query synthesis module.

3.2.5. Query synthesis

In sections 3.2.2 to 3.2.4, various processes required to obtain the term level information for query synthesis were described. The algorithms implemented in the following sub sections were developed by Patro. In order to perform query synthesis based on feedback information gathered, the following steps need to be undertaken in order:

- Select terms from the feedback that will be used in the query synthesis based on their coverage
- Form a Boolean expression that rejects all irrelevant documents
- Intermediate query optimization
- Final query synthesis

The main contribution of this work is to implement the algorithms in a real-time context using concurrent threads. The following sub sections briefly describe the query synthesis algorithm.

3.2.5.1. Selecting terms for query synthesis based on coverage

The words obtained from the feedback marked documents have different frequency of occurrence within the document. The coverage of a term among the documents marked in the feedback process is defined as the ratio of the number of documents selected by the term to the total number of documents marked by feedback. Figure 3-5 shows the formula used to define coverage of a term.

$$\text{Coverage of term} = \frac{\text{Number of documents matched by term}}{\text{Total number of documents marked for feedback}}$$

Figure 3-5. Formula for coverage

If few terms had a very high frequency among all the documents retrieved, then the resulting query containing that term would match both relevant and irrelevant documents. Similarly, terms that match very few documents are not capable of improving the search performance due to their limited coverage in the feedback documents. The range of coverage values selected is from 20% to 60%, where 20% corresponds to terms that have lesser coverage and 60% to terms that have more coverage.

3.2.5.2. Form a Boolean expression that rejects all irrelevant documents

As the next step in the query synthesis procedure, a Boolean expression is constructed that selects every relevant document and rejects all irrelevant documents. This Boolean expression is called the Conjunctive Normal Form (CNF). The buildCNF algorithm, designed by Patro (2006, p. 36) is used to perform this task.

The algorithm forms certain data structures called maxterms. The maxterm selects every document in the relevant document set and rejects one or more irrelevant documents. By careful selection of maxterms, the algorithm produces a list of maxterms that effectively rejects every document in the irrelevant document set. The conjunction of these maxterms is the required Boolean expression.

The maxterms are formed by selecting one term at a time from the list of terms. This list of terms is chosen from a sorted list based on their selectivity value, which gives higher priority to terms that have potential to select more relevant documents. The formula for computing selectivity for a term in the maxterm is defined in Figure 3-6.

$$\text{Selectivity of term in maxterm} = \frac{(R_t \times IR_t)}{((R_r+1) \times (IR_r+1))}$$

where

R_t = number of relevant documents selected by term t and not selected by any term in the current maxterm formed.

IR_t = number of irrelevant documents that term t does not select among those that are selected by previous maxterms and not selected by terms in the current maxterm formed.

R_r = number of relevant documents that remain unselected after adding term t to the current maxterm formed.

IR_r = number of irrelevant documents that term t selects among those that are selected by the previous maxterms and not selected by terms in the current maxterm formed.

Figure 3-6. Selectivity of term

The terms in the maxterm are sorted and joined together by the OR operator "|". The maxterm formed increasingly reject the remaining irrelevant documents while selecting all relevant documents. The combined list of maxterms rejects all irrelevant documents (while selecting all relevant documents) and can be joined together in a Boolean expression. Using this Boolean expression, the next step of optimizing the intermediate query can be done.

3.2.5.3. Intermediate query optimization

The Boolean expression formed in the previous subsection may be too large for direct use in the search engine. The Boolean expression currently in CNF can be converted into Disjunctive Normal Form (DNF) by applying the distribution property: A (B | C) ≡ (A B) | (A C)

The resulting set of terms is called minterms. The equivalence of the above relation means that each minterm would reject an irrelevant document. The entire set of

minterms may select all relevant documents, but of this, several minterms do not select any relevant document. These redundant minterms are removed at a later stage (see Section 3.2.5.4).

The quality of the minterms is calculated based on the ratio of number of relevant documents that the minterm selects to the number of irrelevant documents selected (Patro 2006). The minterms and their quality values are passed onto the next module.

3.2.5.4. Final query synthesis

The final module constructs the query based on the quality values for minterms computed in the previous section. This query synthesis is performed over a configurable number of threads to give multiple queries concurrently. In the current implementation, three threads are used for this purpose. The resulting queries can then be selected based on the best fit for the search engine.

Within each thread, the following steps take place:

a. The process starts with the minterms of the highest possible quality. Since most minterms have a quality value of less than 10, a very large value was used initially.

b. If no minterms are found to satisfy the current quality, then the level is reduced to the next available quality level.

c. When the minterm set has been obtained, the next step is to remove the dominated minterms. The dominated minterms select fewer relevant documents than other minterms and thus, can be replaced by those minterms.

d. From this reduced set of minterms, a query is formed using the minimum number of minterms required to select all the relevant documents.

e. The newly formed query is checked for an acceptable length (less than 15 terms). If the query does not satisfy the determined query length, then the process continues from step b with a lower quality level.

At the end of the final query synthesis, the shortest query is selected from the queries produced by multiple threads. This newly formed query is then sent to the user interface module, where it retrieves the search results by communicating with the search engine.

3.3. Summary

The chapter described the design and implementation of a Real-Time query synthesiser, developed to test the real time performance of the query synthesis algorithm proposed by Sunanda Patro. A discussion of the external components used in the system was followed by a description of the software architecture. The final system produced is capable of receiving user feedback, storing the feedback information efficiently in memory and performing query synthesis in an acceptable amount of time. The performance of the Real-Time query synthesiser is discussed in the following chapter.

4. Results and Discussion

The real time query synthesiser can now be used to perform queries as per the topics listed in previous survey (Patro 2006). The aim of this chapter is to establish the effectiveness of the RT query synthesiser as an effective and usable tool for web search. To this goal we propose the following hypothesis:

> *Real time performance can be obtained when performing query synthesis using feedback on web search results*

To establish the hypothesis, we shall report results from search experimentations using the RT query synthesiser for topics previously used by Patro in her user surveys. These experimentations would provide comparison to support the hypothesis.

The comparisons can be broken into three main themes:

- Search performance
- Search session time
- User interface for search

The chapter is organised on the above themes as follows. Section 4.1 describes the experimental set up and related procedural details. Section 4.2 evaluates the search performance metrics, precision and coverage, for the present implementation against those reported earlier. Section 4.3 compares search session times for various information needs to establish that the RT synthesiser can be used to perform searches in times commensurate with the typical web search and far smaller than those reported in the user surveys by Patro. Section 4.4 describes the user interface and its suitability for relevance feedback entry. Section 4.5 provides a summary for the chapter.

4.1. Description of the experimental setup

The intention of these experiments are to prove that when using feedback on web search results, it is possible to obtain real-time performance with time delays comparable to those experienced in a normal interaction with a web search engine. To achieve this, the search evaluation closely follows the survey conducted in previous research by Patro to compare with the search performance indicators of her offline synthesis method.

A new user survey was not conducted because by closely following the topic descriptions detailed in Patro's survey, the results obtained from the current software could be compared with the values reported by her offline synthesis algorithm. The search results obtained from user sessions in the previous survey were likely to be almost similar for the topics defined. In addition to ascertaining that the current implementation provides similar precision and recall performance, the new metric that is to be verified is the session duration that searchers experience as they provide feedback. Session durations, similar to those experienced by a typical Google web searcher, are considered acceptable. This data can be easily obtained by experimentation and does not require extensive independent user sessions. A user survey would have been required if the search performance of the query synthesis software was to be tested for a larger number of topics, than the original 25.

The test environment for the conducting the experiments were on a machine of the following configuration:

- Processor: Pentium IV 1.6 GHz
- Memory: 1GB RAM

Since the current experiments did not involve an independent user survey, the tests were conducted on the same topics as the previous survey reported by Patro. To conduct the tests, the following steps were carried out during the search sessions:

- Topic was viewed along with its detailed description
- Initial one-word query was entered as indicated against the topic name

- From the search results displayed, the relevant/irrelevant results were marked either based on search result summary or by viewing the web page in a browser.
- After feedback for the first 20 documents were marked, the process for query synthesis was started.
- From the search results returned based on the newly synthesised query, the results were re-evaluated for the top 20 documents.

By using the same initial query, it was possible to obtain search results closely related to the result set obtained in Patro's survey. The refined search results were evaluated and its search metrics recorded.

The following observations were recorded for search results using the one-word queries to characterise search prior to query synthesis (these recorded values are shown to be similar to those done by Patro):

- Time taken to indicate feedback on top 20 search results
- Number of relevant documents found in the first 10 documents
- Number of relevant documents found in the first 20 documents
- Number of irrelevant documents found in the first 20 documents
- Total number of documents matched by the initial query

The observations made prior to query as noted above are displayed in Appendix 1. After the query synthesis is done, the new search result set is evaluated based on the following search attributes:

- Synthesised query produced by the software
- Time taken for the total search process (including time taken for query synthesis)
- Time taken for the software to produce the synthesised query
- Number of relevant results in the top 10 documents
- Number of relevant results in the top 20 documents

- Total number of documents matched by the synthesised query

For the purposes of comparing search performance, it was necessary to use the topics used in the previous user survey and compare the search metrics. Guidelines providing detailed description of the topics were used previously to describe the information need and these were closely consulted when conducting data collection.

The outcome from the data collection experiment along with the generated synthesised queries is shown in Appendix 2.

4.2. Evaluation of search performance

To compare performances of the different queries, certain performance metrics need to be used. For evaluation of search results in text retrieval, precision and recall have historically been the metrics used (Harman 1992). But in web information retrieval, these metrics needs to be modified in the light of the nature of the search space.

4.2.1. Precision

Precision is defined easily as the number of relevant documents present in the documents viewed. Of the number of documents viewed, if 20% of the documents are relevant, then the precision of the search is said to be 0.2. This indicates to the user the ease of finding the relevant documents in the first few pages of the results. For most search users, precision is the search performance metric that they evaluate most of their searches on.

For purposes of this experiment, the search sessions involved using feedback obtained on the first 20 documents and hence the precision indicated is calculated based on the top 20 documents. Table 4-1 provides query performance in terms of precision from user sessions, those reported by Patro and those obtained during our current experiment using real-time query synthesiser.

Topic	User synthesis	Offline synthesis	Real Time synthesis
Amoeba	0.8	1.0	0.8
Angle	0.8	0.95	0.45
Ashoka	0.55	0.6	0.6
Cobra	0.5	0.9	0.7
Eucalyptus	0.75	0.85	0.95
Graphite	0.55	0.85	0.5
Grasshopper	0.7	1.0	0.75
Igloo	0.65	0.8	0.55
Kangaroo	0.7	0.9	1.0
Kohinoor	0.55	0.9	0.9
Konark	0.85	1.0	1.0
Lava	0.75	0.95	0.9
Leopard	0.7	1.0	0.45
Lotus	0.7	0.9	0.8
Mango	0.95	0.95	0.65
Nile	0.75	0.75	0.4
Ostrich	0.85	0.9	0.75
Ozone	1.0	1.0	0.75
Pyramid	0.9	1.0	0.45
Radium	1.0	1.0	0.75
Rainbow	0.75	1.0	0.05
Sun	0.8	0.85	0.7
Titanic	0.6	0.95	0.85
Veena	0.75	0.85	1.0
Vitiligo	1.0	1.0	0.95

Table 4-1. Comparison of search performance based on precision across topics

The precisions values in the table suggest that the real-time query synthesiser could perform at all levels similar to those of users in previous survey sessions. As reported by Patro, these users are typically experienced web searchers and needed long sessions to obtain the reported search performance. The precision obtained in this experiment is slightly inferior to those reported for Patro's algorithms. This, however, is accounted for by the fact that we used only feedback on 20 documents as opposed to many more documents in query synthesis by Patro (50 or more).

4.2.2. Recall

Defining the search performance metric - recall, for web collections is more challenging than defining precision. To the web search user, this indicates the coverage or range of relevant documents that the query could access. The total number of documents suggested by the search engine is not an estimate of the

relevant documents because of the documents retrieved, many will not be relevant to the information need.

For web collections, obtaining the exact number of relevant documents currently indexed by the search engine is a difficult task. This can be attributed to the difficulty in discerning relevance of a large number of documents (in the order of 100000's) in an automatic manner.

For the purposes of comparing recall in web collections, Patro had developed a metric that estimated the approximate percentage of relevant documents covered based on the precision for the first 20 documents. This formula is used in the current experiment to compare with recall values obtained from the previous results. The formula for computing recall (Patro 2006) is shown in Figure 4-1.

$$\textbf{Recall} = \textbf{min}(\ \textbf{round}(\log_2(x*y)/20,2),\ 1)$$

where

x = total results returned for the query

y = precision for top 20 documents

Figure 4-1. Formula for computing recall for web collections

The formula for recall as shown above estimates the number of relevant documents that are present in the result set from the number of relevant documents present in the first 20 documents.

For purposes of this experiment, the search sessions involved using feedback obtained on the first 20 documents and hence the recall indicated is calculated based on the top 20 documents. Table 4-2 provides query performance in terms of recall from user sessions, those reported by Patro and those obtained during our current experiment using real-time query synthesiser.

In general, the recall values obtained from the experiments for real time synthesis were as good as the recall values recorded previously. It can be noted that in majority of the cases, the recall values perform better than those reported by the offline query synthesis.

Topic	User synthesis	Offline synthesis	Real Time synthesis
Amoeba	0.63	0.74	**0.85**
Angle	0.97	1.0	**1.0**
Ashoka	0.56	0.71	**0.74**
Cobra	0.61	0.64	**0.89**
Eucalyptus	0.79	0.84	**0.93**
Graphite	0.46	0.83	**0.9**
Grasshopper	0.7	0.73	**0.96**
Igloo	0.63	0.68	**0.76**
Kangaroo	0.83	0.68	**0.7**
Kohinoor	0.47	0.63	**0.81**
Konark	0.69	0.63	**0.42**
Lava	0.76	0.9	**0.97**
Leopard	0.62	0.65	**0.81**
Lotus	1.0	0.98	**1.0**
Mango	0.7	0.94	**0.71**
Nile	0.82	0.95	**1.0**
Ostrich	0.6	0.7	**0.89**
Ozone	0.77	0.95	**1.0**
Pyramid	0.88	0.93	**0.94**
Radium	0.73	0.87	**0.79**
Rainbow	0.91	0.97	**0.77**
Sun	1.0	1.0	**0.82**
Titanic	0.65	0.72	**0.82**
Veena	0.61	0.64	**0.94**
Vitiligo	0.66	0.75	**0.91**

Table 4-2. Comparison of search performance based on recall across topics

Recall and precision often provide divergent views of the queries. The metrics are often combined to form a single metric for better comparison of search performance.

4.2.3. F1 Measure

F1 measure provides a balanced search performance metric that is calculated as the harmonic mean of the precision and recall values. This metric allows for cases where the precision and recall values are of varying qualities, but their combined effect would give a better estimate of their impact on search performance.

Table 4-3 provides query performance in terms of F1 measure from user sessions, those reported by Patro and those obtained during our current experiment using real-time query synthesiser.

Topic	User synthesis	Offline synthesis	Real Time synthesis
Amoeba	0.7	0.85	**0.82**
Angle	0.88	0.97	**0.62**
Ashoka	0.55	0.65	**0.66**
Cobra	0.55	0.75	**0.81**
Eucalyptus	0.77	0.84	**0.91**
Graphite	0.5	0.84	**0.64**
Grasshopper	0.7	0.84	**0.84**
Igloo	0.64	0.74	**0.64**
Kangaroo	0.76	0.77	**0.82**
Kohinoor	0.51	0.74	**0.85**
Konark	0.76	0.77	**0.59**
Lava	0.75	0.92	**0.93**
Leopard	0.66	0.79	**0.58**
Lotus	0.82	0.94	**0.89**
Mango	0.81	0.94	**0.76**
Nile	0.78	0.84	**0.57**
Ostrich	0.7	0.79	**0.81**
Ozone	0.87	0.97	**0.86**
Pyramid	0.89	0.96	**0.61**
Radium	0.84	0.93	**0.77**
Rainbow	0.82	0.98	**0.09**
Sun	0.89	0.92	**0.76**
Titanic	0.62	0.82	**0.83**
Veena	0.67	0.73	**0.97**
Vitiligo	0.8	0.86	**0.93**

Table 4-3. Comparison of search performance based on F1 measure across topics

The comparison of F1 measure shows that the combined effect of precision and recall are almost as good as those produced by offline synthesis procedure, and definitely much better than user modified queries in most cases. The occasional drop in search performance could be because of the very few relevant documents that could be found related to the topic when initially indicating feedback. This would have in turn affected the ability of the real-time query synthesiser to produce a query of appropriate precision and recall.

4.3. Evaluation of Search session times

The session time of the software is sum of the time taken for web search user to enter feedback, time taken for the real time synthesiser to produce the synthesised query and the time taken to display the new search results page. Time taken to indicate feedback on search results vary with topics and the complexity of search results returned.

Table 4-2 provides query performance in terms of session times from user sessions, those reported by Patro and those obtained during our current experiment using real-time query synthesiser.

		Real Time synthesis	
Topic	User Synthesis (minutes)	Feedback entry time (Seconds)	Synthesis time (Seconds)
Amoeba	35	90	2
Angle	40	95	3
Ashoka	40	85	1
Cobra	30	70	1
Eucalyptus	30	85	1
Graphite	30	105	2
Grasshopper	25	55	2
Igloo	20	70	1
Kangaroo	20	105	2
Kohinoor	20	78	2
Konark	10	115	2
Lava	35	90	1
Leopard	35	115	1
Lotus	10	80	21
Mango	40	90	2
Nile	15	70	2
Ostrich	20	80	40
Ozone	30	70	1
Pyramid	20	100	1
Radium	20	75	10
Rainbow	20	55	1.5
Sun	15	50	1.5
Titanic	10	80	2.5
Veena	10	80	1
Vitiligo	35	90	1
Averages	**24.6**	**83.12**	**4.34**

Table 4-4. Comparison of search session times across topics

Users in Patro's survey took on average 24 minutes to modify and finalize a query as their best effort, whereas the real time synthesis was able to present a query in less than 90 seconds. This includes the time spent by the search user in evaluating the relevance of results on the top 20 documents. This is a significantly small fraction of the time taken by any of the methods detailed earlier.

The small amount of time taken for synthesis gives rise to greater flexibility in performing multiple feedback iterations. By giving additional input to the query synthesis engine, search performance can be improved even further.

4.4. User interface

The simple search engine interface, introduced by Google, has become a de facto standard to present search results (Sinha 2005). This can be attributed to a well presented format of displaying the search results, with easy navigation and links to external websites. The data-driven approach has helped in enhancing the search experience of the users, though web-design experts have a different opinion (Boeving 2006).

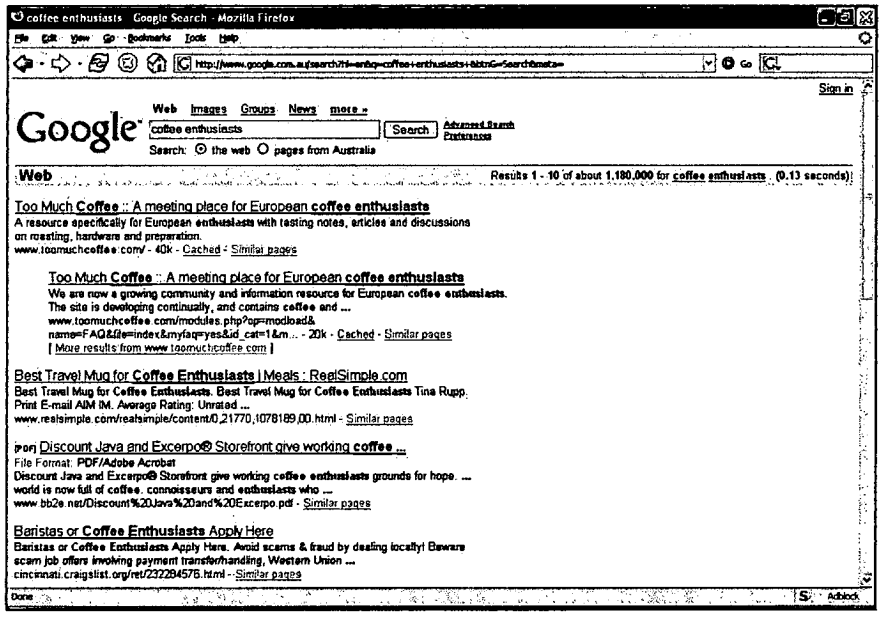

Figure 4-2. Search engine interface – Google

Since the interface is fairly ubiquitous across search engines, providing an interface that is uncluttered and suitable for quick indication of relevance feedback is necessary. This is made possible by adding a feedback radio button to the right corner of the each individual search result as shown in Figure 4-3.

The feedback radio buttons positioned next to the search results provides a neat way to specify feedback. After the feedback has been entered, quick evaluation of the feedback entered can be done before sending the marked results to the query synthesiser engine. The search results in Figure 4-3 have been highlighted by the use of green ticks for relevant and red crosses for irrelevant documents.

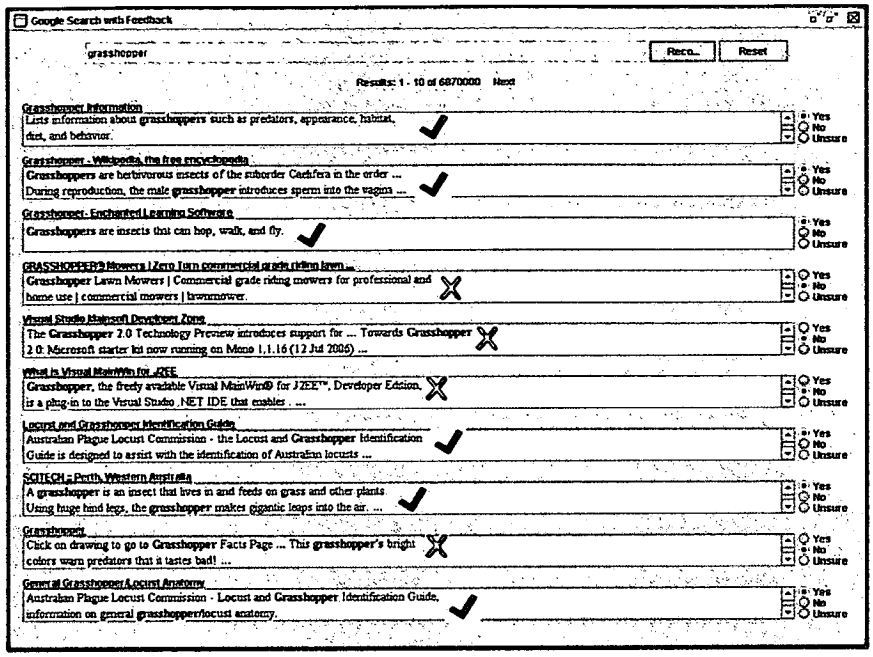

Figure 4-3. Real time synthesiser user interface after feedback entry

After the feedback is sent to the query synthesiser, the updated search result list is loaded into a similar screen format, as shown in Figure 4-4. The feedback radio buttons have been retained in case additional feedback iterations are carried out. The synthesised query appears in the search box. The relevant results returned by the search engine have been marked with green ticks for better comprehension.

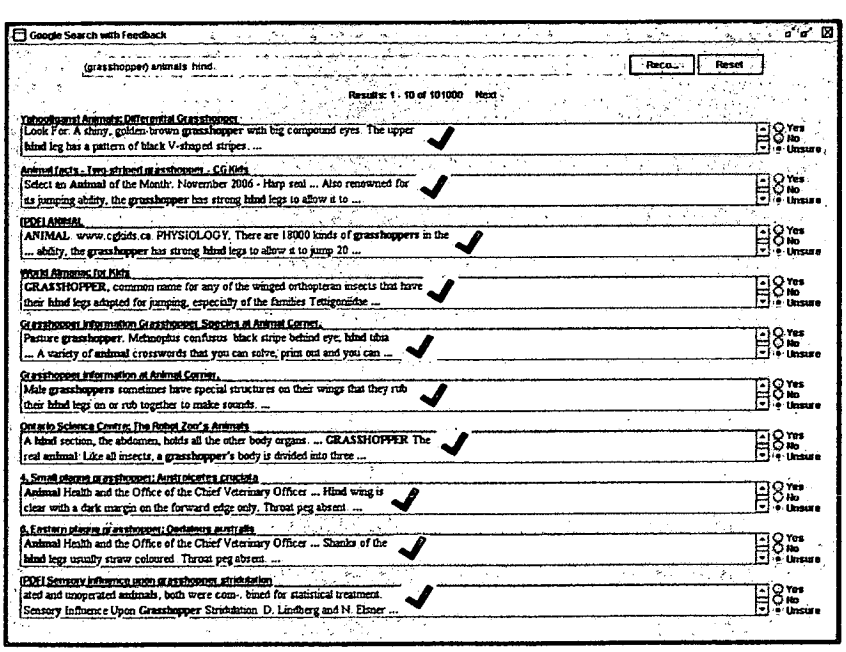

Figure 4-4. Real time synthesiser user interface after new search results displayed

As is evidenced from the discussion above, the software provides a clean and uncluttered user interface for marking feedback on search results. By minimal additions to a user interface commonly seen in commercial web search engines, the interface allows quick learning and makes it easier for search users to effectively indicate their feedback.

4.5. Conclusion

The experiments have established that real time query synthesiser can provide good search performance, with very acceptable search times and easily accessible user interface. These results show that real time performance can be achieved when performing query synthesis using feedback on web search results. The final chapter outlines future work and summarizes our results.

5. Future work and Conclusion

The real-time query synthesising system has been implemented tested and verified for search quality in addition to desired non-functional requirements - real-time performance and user interface. However, to keep the development free of extraneous constraints, the present implementation is on a computer platform with a liberal amount of memory. Other resources like processing speed, disk space and monitor are modest by current hardware standards. There may be many potential users of this system, who may have more limited computer configuration and resources available. A possible direction of continued work on this software may attempt to give implementations that satisfactorily function on even these platforms. Also, the current system has not been extensively trialled by a wide group of users. Such user surveys can provide further inputs to tune and customize the software for more streamlined performance.

The current system is loosely integrated with the browser and serves the function of a document viewer, when required, during the feedback entry process. A completely web based system would allow a seamless and responsive user interface which could make use of emerging web technology - AJAX (Asynchrononous JavaScript and XML). The recently introduced Google AJAX Search API could provide as a starting point for systems development, if the API search results are as good as those obtained on the web version.

In the current system, the results reported were based on feedback received on the first 20 documents. This was found to give good search performance when compared to the results reported in the previous survey. The effects on search performance can be investigated when the number of documents used for feedback varies to 30, 40 or even 10.

5.1. Conclusion

The thesis investigated the real-time performance of web query synthesis when using relevance feedback information from search users. To provide a recap of the thesis, the chapter themes are summarised. The introduction outlined the goals that were to be achieved by this work. The literature review explored the research done in web information retrieval and what characteristics make it different from the traditional text retrieval. The research done by Sunanda Patro in query synthesis using user feedback was also presented. This was followed by a discussion of the system architecture of the Real-Time query synthesiser and its implementation, developed to test the hypothesis. The Results and Discussion chapter presented the hypothesis and discussed the search performance of the implemented system in real-time to validate the hypothesis.

The following goals were achieved using the software:

- The search performance achieved by the real-time query synthesiser is similar to that reported in the Patro's survey, in terms of precision and recall. The combined search performance metric, F1 measure, showed a balanced view of search performance and confirmed the similarity
- The time taken for query synthesis is similar to the time taken for a search session with a web search engine. These times were found to be significantly lower than those reported for user sessions in Patro's survey.
- The user interface is intuitive and feedback entry can be done in an easy manner.

The Real-Time query synthesiser is thus an effective tool to perform query synthesis when incorporating user feedback on web search results.

References

Alemayehu, N 2003, 'Analysis of performance variation using query expansion', *J. Am. Soc. Inf. Sci. Technol.*, vol. 54, no. 5, pp. 379-91.

Arasu A., Cho J., Garcia-Molina H., Paepcke A. & Raghavan S. 2001, 'Searching the Web', *ACM Trans. Inter. Tech.*, vol. 1, no. 1, pp. 2-43.

Baeza-Yates, R & Ribeiro, BdAN 1999, *Modern information retrieval*, Addison-Wesley, Harlow.

Bharat, K & Henzinger, MR 1998, 'Improved algorithms for topic distillation in a hyperlinked environment', in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, Melbourne, Australia, pp. 104-11.

Boeving, A 2006, *Interface Design: The uglier the better?*, viewed 22 November, 2006 <http://clearfunction.com/weblog/interface-design-the-uglier-the-better>.

Brin, S & Page, L 1998, 'The anatomy of a large-scale hypertextual Web search engine', in *Proceedings of the seventh international conference on World Wide Web 7*, Elsevier Science Publishers B. V., Brisbane, Australia, pp. 107-17.

Broder, A 2002, 'A taxonomy of web search', *SIGIR Forum*, vol. 36, no. 2, pp. 3-10.

Bruza, P, McArthur, R & Dennis, S 2000, 'Interactive Internet search: keyword, directory and query reformulation mechanisms compared', in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, Athens, Greece, pp. 280-7.

Chakrabarti, S 2003, *Mining the Web : discovering knowledge from hypertext data*, Morgan Kaufmann series in data management systems, Morgan Kaufmann, Amsterdam ; Boston.

Chowdhury, GG 2004, *Introduction to modern information retrieval*, 2nd edn, Facet Publishing, London.

Chung, Y-M, He, Q, Powell, K & Schatz, B 1999, 'Semantic indexing for a complete subject discipline', in *Proceedings of the fourth ACM conference on Digital libraries*, ACM Press, Berkeley, California, United States, pp. 39-48.

Google 2005, *Google Help: Basics of Search*, viewed 29 November, 2006 <http://www.google.com/help/basics.html>.

Google 2006, *Google SOAP Search API (Beta)*, viewed 28 November, 2006 <http://code.google.com/apis/soapsearch/index.html>.

Harman, D 1992, 'Relevance feedback revisited', in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, Copenhagen, Denmark, pp. 1-10.

Hassel, M 2006, *Java package for working with Random Indexing and Granska*, viewed 22 November, 2006 <http://www.nada.kth.se/~xmartin/>.

HTML Parser, 2006, viewed 22 November, 2006 <http://sourceforge.net/projects/htmlparser>.

Jansen, BJ, Spink, A, Bateman, J & Saracevic, T 1998 'Real life information retrieval: a study of user queries on the Web ', *SIGIR Forum* vol. 32 no. 1 pp. 5-17

Kobayashi, M & Takeda, K 2000, 'Information retrieval on the web', *ACM Comput. Surv.*, vol. 32, no. 2, pp. 144-73.

Lawrence, S & Giles, CL 1998, 'Searching the World Wide Web', *Science*, vol. 280, no. 5360, pp. 98-100.

Leroy, G, Lally, AM & Chen, H 2003 'The use of dynamic contexts to improve casual internet searching ', *ACM Trans. Inf. Syst.* , vol. 21 no. 3 pp. 229-53

Liu, X & Croft, WB 2004, 'Cluster-based retrieval using language models', in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, Sheffield, United Kingdom, pp. 186-93.

Lucas W. & Topi H. 2002, 'Form and Function: the impact of query term and operator usage on web search results', *J. Am. Soc. Inf. Sci. Technol.*, no. 53(2), pp. 95-108.

Muramatsu, J & Pratt, W 2001, 'Transparent Queries: investigation users' mental models of search engines', in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, New Orleans, Louisiana, United States, pp. 217-24.

Patro, S 2006, 'Synthesising Web Search Queries from Example Text Documents', Research Masters thesis, University of Tasmania.

Porter, M 2005, *The Porter Stemming Algorithm*, viewed 22 November, 2006 <http://www.tartarus.org/~martin/PorterStemmer/index.html>.

Porter, MF 1997, 'An algorithm for suffix stripping', in *Readings in information retrieval*, Morgan Kaufmann Publishers Inc., pp. 313-6.

Robertson, SE, Rijsbergen, CJv & Porter, MF 1981, 'Probabilistic models of indexing and searching', in *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, Butterworth \& Co., Cambridge, England, pp. 35-56.

Roubtsov, V 2002, *Java Tip 130: Do you know your data size?*, JavaWorld.com, viewed 22 November, 2006 <http://www.javaworld.com/javaworld/javatips/jw-javatip130.html>.

Ruthven, I & Lalmas, M 2003, 'A survey on the use of relevance feedback for information systems', *Knowledge engineering Review*, vol. 18, no. 2, pp. 95-145.

Salton 1984, 'The use of extended Boolean logic in information retrieval', in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, ACM Press, Boston, Massachusetts, pp. 277-85.

Salton, G 1971, *The SMART Retrieval System:Experiments in Automatic Document Processing*, Prentice-Hall, Inc.

Salton, G, Fox, EA & Wu, H 1983, 'Extended Boolean information retrieval', *Commun. ACM*, vol. 26, no. 11, pp. 1022-36.

Sinha, R 2005, *Google's pragmatic, data-driven approach to user interface design*, viewed 29 November, 2006 <http://www.rashmisinha.com/archives/05_01/googles-pragmatic-datadriven-approach-to-user-interface-design.html>.

Spink, A & Jansen, BJ 2004, *Web search : public searching on the web*, Information science and knowledge management ; v. 6, Kluwer Academic Publishers, Dordrecht, Netherlands ;.

Sun Microsystems 2004a, *Hashtable (Java 2 Platform SE 5.0)*, <http://java.sun.com/j2se/1.5.0/docs/api/java/util/Hashtable.html>.

Sun Microsystems 2004b, *Vector (Java 2 Platform SE 5.0)*, <http://java.sun.com/j2se/1.5.0/docs/api/java/util/Vector.html>.

Sun Microsystems 2004c, *HashSet (Java 2 Platform SE 5.0)*, <http://java.sun.com/j2se/1.5.0/docs/api/java/util/HashSet.html>.

Van Rijsbergen, CJ 1983, 'Information retrieval: new directions: old solutions', in *Proceedings of the 6th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, Bethesda, Maryland, pp. 264-5.

Van Rijsbergen, CJ 1986, 'A new theoretical framework for information retrieval', in *Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, Palazzo dei Congressi, Pisa, Italy, pp. 194-200.

Zhao, R & Grosky, WI 2002 'Bridging the semanitic gap in image retrieval ', in *Distributed multimedia databases: techniques \& applications* Idea Group Publishing, pp. 14-36

Bibliography

Buckley, C, Salton, G & Allan, J 1994, 'The effect of adding relevance information in a relevance feedback environment', in Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, Springer-Verlag New York, Inc., Dublin, Ireland, pp. 292-300.

Chowdhury, GG 2004, Introduction to modern information retrieval, 2nd edn, Facet Publishing, London.

Cohen, WW & Singer, Y 1996, 'Learning to query the Web', in AAAI-96 Workshop on Internet-Based Information Systems, AAAI Press, Menlo Park, CA, pp. 16-25.

Das-Gupta, P 1988, 'Rough sets and information retrieval', in Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, Grenoble, France, pp. 567-81.

Eastman, CM 2002, '30,000 hits may be better than 300: precision anomalies in internet searches', J. Am. Soc. Inf. Sci. Technol., vol. 53, no. 11, pp. 879-82.

Frakes, WB & Baeza-Yates, R 1992, Information retrieval : data structures and algorithms, Prentice-Hall, Englewood Cliffs, N.J.

Harman, D 1988, 'Towards interactive query expansion', in Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, Grenoble, France, pp. 321-31.

Johnson, D, Malhotra, V, Vamplew, P & Patro, S 2004, 'Refining Search Queries from Examples Using Boolean Expressions and Latent Semantic Analysis', paper presented to AISAT2004: The 2nd International Conf. on Artificial Intelligence in Science and Technology, Hobart.

Khan, MS & Khor, S 2004, 'Enhanced web document retrieval using automatic query expansion', J. Am. Soc. Inf. Sci. Technol., vol. 55, no. 1, pp. 29-40.

Kherfi, ML, Ziou, D & Bernardi, A 2002, 'Learning from negative example in relevance feedback for content-based image retrieval.' in Proceedings of the IEEE International Conference on Pattern Recognition (ICPR), Québec, Canada).

Magennis, M & Van Rijsbergen, CJ 1997, 'The potential and actual effectiveness of interactive query expansion', in Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, Philadelphia, Pennsylvania, United States, pp. 324-32.

Malhotra, V, Patro, S & Johnson, D 2005, 'Synthesise web queries: Search the Web by examples', paper presented to 7th International Conference on Enterprise Information Systems (ICEIS2005), Volume 2, Miami, Florida, 24-28 May 2005.

Mozilla Corporation 2006, Integrated Search, viewed 12 May 2006 <http://www.mozilla.com/firefox/search.html>.

Patro, S & Malhotra, V 2005, 'Characteristics of the Boolean web search query: Estimating success from characteristics', paper presented to 1st international conf. on web info. systems and technologies (WEBIST2005), Miami, Florida, 26-28 May 2005.

Pawlak, Z, Grzymala-Busse, J, Slowinski, R & Ziarko, W 1995, 'Rough sets', Commun. ACM, vol. 38, no. 11, pp. 88-95.

Ruthven, I 2003 'Re-examining the potential effectiveness of interactive query expansion ', in Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval ACM Press, Toronto, Canada pp. 213-20

Shen, X & Zhai, C 2005, 'Active feedback in ad hoc information retrieval', in Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, Salvador, Brazil, pp. 59-66.

Singh, S & Dey, L 2005, 'A rough-fuzzy document grading system for customized text information retrieval', Inf. Process. Manage., vol. 41, no. 2, pp. 195-216.

White, RW, Ruthven, I & Jose, JM 2005, 'A study of factors affecting the utility of implicit relevance feedback', in Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, Salvador, Brazil, pp. 35-42.

Witten, IH, Moffat, A & Bell, TC 1999, Managing gigabytes : compressing and indexing documents and images, 2nd edn, Morgan Kaufmann series in multimedia information and systems, Morgan Kaufmann Publishers, San Francisco, Calif.

Wong, SKM & Ziarko, W 1986, 'A machine learning approach in information retrieval', in Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, Palazzo dei Congressi, Pisa, Italy, pp. 228-33.

Yoshioka, M & Haraguchi, M 2005, 'On a combination of probabilistic and boolean ir models for WWW document retrieval', ACM Transactions on Asian Language Information Processing (TALIP), vol. 4, no. 3, pp. 340-56.

Zadeh, LA 1994 'Fuzzy logic, neural networks, and soft computing ', Commun. ACM vol. 37 no. 3 pp. 77-84

Appendices

Appendix 1. Information recorded prior to query synthesis

Topic	Feedback entry time (sec)	Relevant in top10 documents	Relevant in top 20 documents	Irrelevant in top 20 documents	Total search pages
Amoeba	90	4	8	12	3110000
Angle	95	4	9	11	128000000
Ashoka	85	4	6	12	1600000
Cobra	70	1	1	16	53700000
Eucalyptus	85	8	10	8	8080000
Graphite	105	4	4	16	22700000
Grasshopper	55	7	8	11	6260000
Igloo	70	2	2	18	7440000
Kangaroo	105	5	8	12	13500000
Kohinoor	78	2	3	14	1480000
Konark	115	8	12	8	262000
Lava	90	3	4	16	23000000
Leopard	115	4	6	14	25100000
Lotus	80	4	4	13	74200000
Mango	90	2	3	17	25900000
Nile	70	6	6	14	20000000
Ostrich	80	5	9	11	5900000
Ozone	70	2	8	8	28500000
Pyramid	100	2	2	18	27800000
Radium	75	6	9	11	5150000
Rainbow	55	2	3	17	91600000
Sun	50	2	2	16	982000000
Titanic	80	3	7	13	20700000
Veena	80	4	5	15	2850000
Vitiligo	90	10	17	1	630000

Appendix 2. Synthesised queries and their search detail

Synthesised queries	Time for search process (sec)	Time taken for synthesis (msec)	Total results after feedback	Relevant in top 10 documents	Relevant in top 20 documents	Chosen for statistic
(amoeba) moving food	2	187	165000	9	16	y
(amoeba) organism rotifer	1.5	93	21800	9	17	
(amoeba) protozoa food	2	78	62500	8	15	
(angle) vertical	3	188	16100000	4	9	y
(ashoka) kalinga	1	172	48400	8	12	y
(ashoka) led	1.5	63	739000	4	9	
(ashoka) violence	1.5	47	448000	2	7	
(cobra) elapids	1.5	93	9410	10	20	
(cobra) french	1	31	1680000	1	2	
(cobra) mythology	1	16	298000	8	15	y
(eucalyptus) bark seed	2.5	234	389000	8	13	
(eucalyptus) & (bark & (related|described))	1	141	334000	10	19	y
(eucalyptus) photo genus	1	156	112000	10	20	
(graphite) conductor	2.5	188	552000	5	10	y
(grasshopper) & (grass|hind)	2.5	172	848000	7	15	y
(grasshopper) animals	2	63	934000	5	14	
(grasshopper) & (hind|animals)	2	46	974000	7	13	
(igloo) required	2	185	325000	7	8	

Synthesised queries	Time for search process (sec)	Time taken for synthesis (msec)	Total results after feedback	Relevant in top 10 documents	Relevant in top 20 documents	Chosen for statistic
(igloo) smaller	2	125	172000	6	10	
(igloo) light entrance	1.5	46	65800	6	11	y
(kangaroo) giganteus & (pouch & (red))	2.5	1047	848	10	20	
(kangaroo) giganteus hind	2	328	16400	10	20	y
(kangaroo) hind & (home & (feet|giganteus)		125547	120000	6	14	
(kohinoor) punjab	2	125	46500	6	12	
(kohinoor) history	2	31	107000	8	13	
(kohinoor) stone	2	31	83500	10	18	y
(konark) & (hindu floral sanctum|temples tradition)	25	23565	364	9	13	
(konark) & (sculptures lake|likely chiseled)	41	407797	51	7	10	
(konark) & (erotic & (feet|floral))	2	297	360	10	20	y
(lava) & (uncommon|surface)	2.5	219	1130000	10	20	
(lava) surface	1.5	125	1840000	9	18	
(lava) slowly	1.5	31	798000	9	18	y
(leopard) goes years	2	406	989000	1	2	
(leopard) east solitary	1	172	165000	9	9	y
(leopard) east females	1	125	588000	10	19	
(lotus) library & (ibm & (training software quickplace))	8	4266	154000	10	20	

Synthesised queries	Time for search process (sec)	Time taken for synthesis (msec)	Total results after feedback	Relevant in top 10 documents	Relevant in top 20 documents	Chosen for statistic
(lotus) ibm resources	21	17859	1800000	6	16	y
(lotus) & (training & (ibm & (resources & (mail))))	2.5	391	989000	10	20	
(mango) related	3	156	1400000	2	4	
(mango) commonly	2	63	482000	8	13	y
(mango) parts	2	31	1100000	7	7	
(nile) & (animals|high|let)	2	328	2720000	5	8	y
(nile) & (high|animals)	1	109	2170000	5	7	
(nile) & (high|let)	1	47	2040000	1	3	
(ostrich) nest & (wild)	2	641	242000	8	14	
(ostrich) months & (north & (females))	40	34765	306000	9	15	y
(ostrich) & (years & (run & (tall))|(fence))	2	422	384000	8	17	
(ozone) protection	2.5	63	5630000	8	17	
(ozone) information & (cancer|need|protection)	1	109	9420000	8	15	y
(ozone) eye	1	265	1160000	5	9	
(pyramid) & (dozen|fed)	1.5	312	1170000	1	3	
(pyramid) archaeologists	1	47	951000	1	5	
(pyramid) egypt story	1.5	47	985000	6	9	y
(radium) isotopes related ra mass	10	8015	78700	7	15	y
(radium) isotopes ra body	2.5	406	80300	10	20	
(radium) & (energy & (related & (number|body)))	2.5	281	419000	3	8	

Synthesised queries	Time for search process (sec)	Time taken for synthesis (msec)	Total results after feedback	Relevant in top 10 documents	Relevant in top 20 documents	Chosen for statistic
(rainbow) & (update frames|recycle examples)	2	344	418000	1	1	
(rainbow) & (parts & (ren|browser))	1.5	94	835000	1	1	y
(rainbow) & (update|examples)	1	31	2870000	1	1	
(sun) animation	1.5	156	10200000	5	9	
(sun) luminous	1.5	63	130000	7	14	y
(sun) gravity	1	47	4380000	10	20	
(titanic) & (deck & (subsequent|task)|(things subsequent))	2.5	437	96600	9	17	y
(titanic) & (recent & (continued|subsequent)|(risk book))	1	140	784000	6	10	
(veena) instrument wood	1.5	296	390000	8	17	
(veena instrument) sitar	1	63	470000	10	20	y
(veena) & (century|instruments)	1	78	458000	10	20	
(vitiligo) disease autoimmune	1.5	172	330000	10	19	y
(vitiligo) disease pernicious	1.5	78	56200	10	20	
(vitiligo) disease home	1	62	436000	8	17	

Appendix 3. List of stopwords used

a
about
above
across
after
afterwards
again
against
all
almost
alone
along
already
also
although
always
am
among
amongst
amoungst
amount
an
and
another
any
anyhow
anyone
anything
anyway
anywhere
are
around
as
at
back
be
became
because
become
becomes
becoming
been
before
beforehand
behind
being
below
beside
besides
between
beyond
bill
both
bottom
but
by
call
can
cannot
cant
co
computer
con
could
couldnt
cry
de
describe
detail
do
done
down
due
during
each
eg
eight
either
eleven
else
elsewhere
empty
enough
etc
even
ever
every
everyone
everything
everywhere
except
few
fifteen
fify
fill
find
fire
first
five
for
former
formerly
forty
found
four
from
front
full
further
get
give
go
had
has
hasnt
have
he
hence
her
here
hereafter
hereby
herein
hereupon
hers
herself
him
himself
his
how
however
hundred
i
ie
if
in
inc
indeed
interest
into
is
it
its
itself
keep
last
latter
latterly
least
less
ltd
made
many
may
me
meanwhile
might
mill
mine
more
moreover
most
mostly
move
much
must
my
myself

name
namely
neither
never
nevertheless
next
nine
no
nobody
none
noone
nor
not
nothing
now
nowhere
of
off
often
on
once
one
only
onto
or
other
others
otherwise
our
ours
ourselves
out
over
own
part
per
perhaps
please
put
rather
re
same
see
seem
seemed
seeming
seems
serious
several
she
should
show
side
since
sincere
six
sixty
so
some
somehow
someone
something
sometime
sometimes
somewhere
still
such
system
take
ten
than
that
the
their
them
themselves
then
thence
there
thereafter
thereby
therefore
therein
thereupon
these
they
thick
thin
third
this
those
though
three
through
throughout
thru
thus
to
together
too
top
toward
towards
twelve
twenty
two
un
under
until
up
upon
us
very
via
was
we
well
were
what
whatever
when
whence
whenever
where
whereafter
whereas
whereby
wherein
whereupon
wherever
whether
which
while
whither
who
whoever
whole
whom
whose
why
will
with
within
without
would
yet
you
your
yours
yourself
yourselves

Appendix 4. List of identifiers using in parsing of Google search page.

HTML Identifier	Code searched for
startPara	
startPara1	<div style=\"margin-left:2.5em\"><
startHTML	<a class=l
startHTML1	div style=\"margin-left:2.5em\"><a class=l
startPDF	[PDF]
startPDF1	<div style=\"margin-left:2.5em\">[PDF]
startDOC	[DOC]
startDOC1	<div style=\"margin-left:2.5em\">[DOC]
endSnippet	
STARTURL	<a class=l href=\"
ENDURL	\">
ENDTITLE	
STARTBODY	
ENDBODY	
STARTBODY1	\">View as HTML
STARTBODY2	 PDF/Adobe Acrobat
STARTPDFURL	PDF/Adobe Acrobat - <a href=\"
STARTPDFTITLE	.pdf\">
ENDPDFTITLE	
STARTDOCTITLE	.doc\">
ENDDOCTITLE	
STARTRESULTS	<font size=-1>Results